

Web Mining with Relational Clustering

Lecture #3: Antonio Gulli

3.1 Introduction and Overview

In this lecture, we review some ideas that are applied in Relational Web Mining. We focus on clustering, classification, keyword extraction, and predictive learning. Relational Web Mining is a very recent discipline motivated by the need to manage *non-numerical objects*, and to describe formally *relations* among them. Examples of relational datasets are Web logs, sequences of Web pages visited by a set of users, and the textual contents of Web pages themselves. In these cases, it is possible to define models where ground objects are non-numeric, and the relations among objects can be defined using non-numerical attributes. These models could potentially be more expressive than non-relational ones, in which objects (such as words) are often mapped into numerical points on a multidimensional space \mathcal{R}^m , and interactions between points are expressed in term of a distance defined on \mathcal{R}^m .

The lecture is organized as follows. In section 3.3 we introduce *non-relational distance-based* methods, and define a notion of similarity among numerical objects. We also introduce the most important clustering algorithms (such as *K*-means and Hierarchical Clustering), some predictive learning techniques, which are used for lazy prediction of function values given a set of already known examples, and some examples of distance based classification.

In section 3.4 we extend these methods to a relational context, giving the notion of *first-order distance* (paragraph 3.5.1) and another kind of relational distance named *edit distance* (paragraph 3.6.1). In particular, in section 3.7.1.1 we present relational predictive learning methodologies and describe RIBL2, a software that uses them. In section 3.7.1.2 we use the first order-metric together with hierarchical agglomerative clustering approach, and present RDBC a software based on these techniques. In section 3.7.1.3 we review how to extend *K*-means to a relational model. Then, in section 3.6 we describe some other relational techniques used to make clustering, keyword extractions and classification in Web domain.

Finally, in section 3.8 we conclude the lecture with our comments and criticism. We suggest other applications where Relational Web Mining could be exploited and we indicate other directions of investigation for future researches.

3.2 Data Mining and Web Mining

Data Mining has become very popular in recent years. Mining techniques are often described as a set of methodologies that allows analyzing a very large data set for extracting and discovering previously unknown structures out of huge amounts of details. Methodologies are used to filter, to cluster, to associate, to rank, to predict, to navigate and to classify information such the “*extracted core*” is more suitable for taking decisions and strategies.

Web mining can be described as the process of structure’s extraction from semi-structured and unlabeled data that contains users’ information (such as access log) or textual information (such as Web pages themselves). There are some similarities and many differences between traditional Data mining and Web Mining. First at all, Web mining very often needs to deal with very large datasets (currently from giga to terabytes). Besides, the huge amount of semi-structured data carries many bad examples, known as outliers. Finally yet importantly, Web objects could be not suitable for being described using numeric vectors. Instead, a relational description could be more accurate.

In next sections, we will introduce several tools that are used to perform Relational data mining and Relational web mining.

3.3 Overview of Non-Relational Mining

The literature describes several non relational mining problems giving a finite set \mathcal{O} of n objects and a finite set \mathcal{V} of m variables which represents proprieties of each object $x \in \mathcal{O}$. More precisely, each x_j , with $1 \leq j \leq m$, can assume a numeric value in a range R^j . Hence, each object can be identified with a given element in the direct product space $\mathcal{U} = \prod_{j=1}^m R^j$. (\mathcal{U} is often called *universe*)

For instance, in order to perform clustering of Web pages contained in a repository P , it is possible to assume that there are at most m distinct words in P . Therefore, each word can be represented using a unique integer `wordID`, and each document $d \in P$ is a represented as a point in \mathcal{R}^m using an m -length bitmap B_d . The bit $B_d[j]$ is set iff the j -th word appears in the document d . Alternatively, one can use $\forall d \in P$ a m -length vectors V_d . $V_d[j]$ represents the number of occurrences of the w_j in each document $d \in P$. These are the well known “bags¹ of words” representations, and the words extracted by the text are named *features*. Therefore, a non-relational approach maps qualitative or categorical (such as red, male) variables to numeric values.

¹“bags” can be thought as sets which allow repetitions

3.3.1 Distance Metric

Several mining methodologies need to measure the dissimilarity among objects by using a distance metric. A *distance function* is a function defined as $dist : \mathcal{O} \times \mathcal{O} \rightarrow [dist_{min}, dist_{max}]$, which satisfies the following:

- $dist(x, x) = dist_{min}$ for all $x \in \mathcal{O}$
- $dist(x, y) = dist(y, x)$ for all $x, y \in \mathcal{O}$ [symmetric relation]

where $0 \leq dist_{min} \leq dist_{max}$ are real numbers or infinity. If $dist_{min} = 0$ and $dist_{max} = \infty$ and it holds the following:

- $dist(x, z) \leq dist(x, y) + d(y, z)$ for all $x, y, z \in \mathcal{O}$ [triangle inequality]
- $dist(x, y) = 0 \rightarrow x = y$ for all $x, y \in \mathcal{O}$

then the distance function is called a *metric distance function*. There are *Euclidean* distances based on position of points in some m -dimensional space, such as the well-known norms ($\|x\|_1 = \sum_{j=1}^m |x_j|$, $\|x\|_2 = \sqrt{\sum_{j=1}^m |x_j|^2}$, and $\|x\|_\infty = \max_j |x_j|$), and *Non Euclidean* distances such as *Jaccard Measure*, defined on two vectors V_{d1} and V_{d2} as the ratio of the size of intersection and union ($dist(V_{d1}, V_{d2}) = 1 - \frac{|V_{d1} \cap V_{d2}|}{|V_{d1} \cup V_{d2}|}$), the *Cosine Measure* defined as the angle between two vectors $dist(V_{d1}, V_{d2}) = \theta = \arccos(V_{d1} \cdot V_{d2} / (|V_{d1}| |V_{d2}|))$, and the *Edit Distance* defined on strings, which we will introduce in section 3.6.1.

3.3.2 Similarity

A similarity function is a function $sim : \mathcal{O} \times \mathcal{O} \rightarrow [s_{min}, s_{max}]$, which satisfies the following:

- $sim(x, x) = sim_{min}$ for all $x \in \mathcal{O}$
- $sim(x, y) = sim(y, x)$ for all $x, y \in \mathcal{O}$ [symmetric relation]

Usually the similarity between two objects $x, y \in \mathcal{O}$ is calculated given a distance metric $dist$. For instance, $sim(x, y) = -\exp^{-dist(x, y)}$ or $sim(x, y) = 1 - dist(x, y)$ that yields values in the $[0, 1]$ range. In fact, similarity measures are typically normalized in the interval of 0 to 1.

Using a mutual distance or a similarity notion, many methodologies used in Data mining can be described in a simple and intuitive way. In the next paragraphs, we review some of these methodologies. Namely, cluster analysis, predictive learning and distance based classification. The interested reader can find more information in [1] and [4].

3.3.3 Cluster Analysis

Cluster Analysis is a mining technique applied to a set of object \mathcal{O} from the universe \mathcal{U} , with a notion d of distance among objects. The goal is *to aggregate* the objects into some numbers of subsets - the clusters -, so that members of a clusters are in some sense maximally similar to each other and are maximally dissimilar to objects belonging to different clusters. There are two different main kinds of approaches:

- **“Hierarchical”**. At the beginning, each point is a cluster itself. Then, the two “closest” clusters are repeatedly merged into one using a bottom-up strategy.
- **“Point Assignment”**. Using this approach, a fixed set of clusters is maintained for each iteration. Objects are iteratively placed into the closest cluster.

Hierarchical approach creates a dendrogram (fig. 3.1), and the user can select the desired number of clusters given in output, by “cutting” the dendrogram at a chosen level of depth.

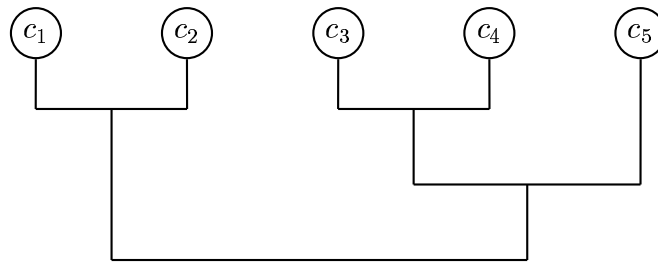


Figure 3.1: An example of dendrogram

Among the point assignment methods, the most known is the *K-means* which assumes a Euclidean space and receives in input the number k of the desired clusters. At the begin, k cluster centers - the centroids - are selected randomly. Then, iteratively (a) each instance is assigned to the nearest cluster center, and (b) a new centroid is computed for each cluster, taking account the just assigned objects. Steps (a) and (b) are repeated in turn until a stopping criterion is satisfied.

There are a plethora of variants both of Hierarchical and Point Assignment, for addressing different aspects of the clustering problem. For instance, there are *hard* clustering problems, where every data record has to belong to one and only one cluster, *soft* clustering, where clusters can overlap, and *fuzzy* clustering where objects belong to each cluster to a certain extent and according to a fuzzy value.

3.3.4 Predictive Learning

Predictive Learning is a mining technique used for *learning* given a set of examples. Objects belong to the universe \mathcal{U} , and there exists the notion of distance between objects. At the beginning, the model assumes that each object is streamed to a repository of available examples \mathcal{E} . During this phase, no computation is performed for learning.

Later on, when the new *query object* $q \in O$ should be classified, it is compared to the already known objects in \mathcal{E} . The result is the prediction of a function value associated with the nearest neighbor among q and the objects in \mathcal{E} . To reduce noise, a common predictive learning variant - named kNN - selects the most frequent prediction given by the set of k -nearest neighbors (using a majority vote mechanism).

Note that during the feeding phase no explicit hypothesis is inferred by the given examples. All computation is just based on a distance metric, and is deferred until a query object is given. This is why predictive learning is considered as a form of lazy computation.

3.3.5 Distance Based Classification

Distance base classification is very similar to predictive learning and it is used for *classifying* unknown data. At the beginning, the repository of examples \mathcal{E} is partitioned according to some pre-selected categories $\mathcal{C} = \{C_1, \dots, C_o\}$ where $\mathcal{C} \subseteq \mathcal{E}$. Then, the notion of distance is used to select the category C_i which minimises the distance among a previously unknown object q and any category in \mathcal{C} . Instead of predicting a function value, the object q is classified as belonging to C_i .

3.4 Overview of Relational Mining

In Relational Mining objects can be described using a set of non-numeric attributes. In this section we will present a model where they are seen as *first order instances* \mathcal{I} taken from an instance space \mathcal{X} . In the next sections, we will present a much simpler model where object are still considered as a set of non-numerical attributes, but they are just seen as “words”. The two models have different domain of applications. The first, was used with success in the genetic domain, where a relational *fine-grain* description of objects can improve the precision reached. The second, was used with success in Web domain where the semi-structured data can be described using a relational model based on words.

3.5 First Order Relational Model

In the first order model, objects are a set of first order instances \mathcal{I} taken from an instance space \mathcal{X} . They are described using a relational language by means of atoms of predicates. An intuitive example is provided in [2]. It is defined a ternary predicate \mathbf{P} describing a particular culinary set with three atoms that represent identifier, price and delivery mode. Each culinary set could contain wine and cheese which are, respectively, described by ground

| | |
|--|--|
| $I = P(\text{set1}, 125, \text{personal})$ cheese(set1, camembert, 150, france) wine(set1, mouton, 1988, 0.75) wine(set1, gallo, 1995, 0.5) vineyard(gallo, famous, large, usa) vineyard(mouton, famous, small, france) | $P(a_1:\text{name}, a_2:\text{num}, a_3:\text{discr})$ cheese($a_1:\text{name}$, $a_2:\text{discr}$, $a_3:\text{num}$, $a_4:\text{discr}$) wine($a_1:\text{name}$, $a_2:\text{discr}$, $a_3:\text{num}$, $a_4:\text{num}$) vineyard($a_1:\text{name}$, $a_2:\text{discr}$, $a_3:\text{discr}$, $a_4:\text{discr}$) |
|--|--|

Figure 3.2: On the left, we see the instance I and the ground atoms which further describe it. On the Right, we have the declarations associated to P and the background predicates “cheese”, “wine”, “vineyard”. Types are in (names, discrete, and numeric).

atoms of a quaternary predicate named **wine** (which represents the set’s identifier, the type, weight and origin of cheese), and **cheese** (which represents the set’s identifier, the origin’s vineyard, the vintage year, and the bottle size). In turn, vineyard is described by a quaternary predicate named **vineyard** (which defines the the vineyard’s identifier, the popularity, the size and the country). Each argument of the predicate symbols can have an associated *type* which describes the set of possible values that it can assume. The relations and types are represented in fig. 3.2.

To have a succinct representation, a set \mathcal{B} of ground atoms - the *background knowledge* - is provided. For a given instance $I \in X$, for a background knowledge \mathcal{B} and for a non-negative integer $d \geq 0$ named *depth bound*, the *case* of \mathcal{I} is defined as the largest set $I \cup \mathcal{B}'$ such that $\mathcal{B}' \subseteq \mathcal{B}$ and for every background atom $B \in \mathcal{B}'$ it holds that there are $d' < d$ and $B_0, B_1, \dots, B_{d'-1}, B_{d'}$ which satisfies (a) $B_0 = I, B_{d'} = B$ and $B_i \in \mathcal{B}'$ for $i = 1, \dots, d' - 1$ (b) B_i and B_{i+1} contain at least one common identifier $\forall i = 0, \dots, d' - 1$. The idea behind the definition is to start with the instance and the background knowledge, and to make recursion within \mathcal{B} up to a maximum depth of d . An example of case computation is given in figure 3.3.

Note that it is intuitive to suppose that the relational model could potentially mine an “information core” better than non-relational one. This is because it allows descriptions of data that are nearest to the real world phenomena. Attributes are not just mapped to points, but they are part of the model and one can define relations among them. Therefore, the model itself shows a larger power of expression. In order to use this model, one needs to extend the notion of distance and to obtain a first-order distance measure. We introduce this extension

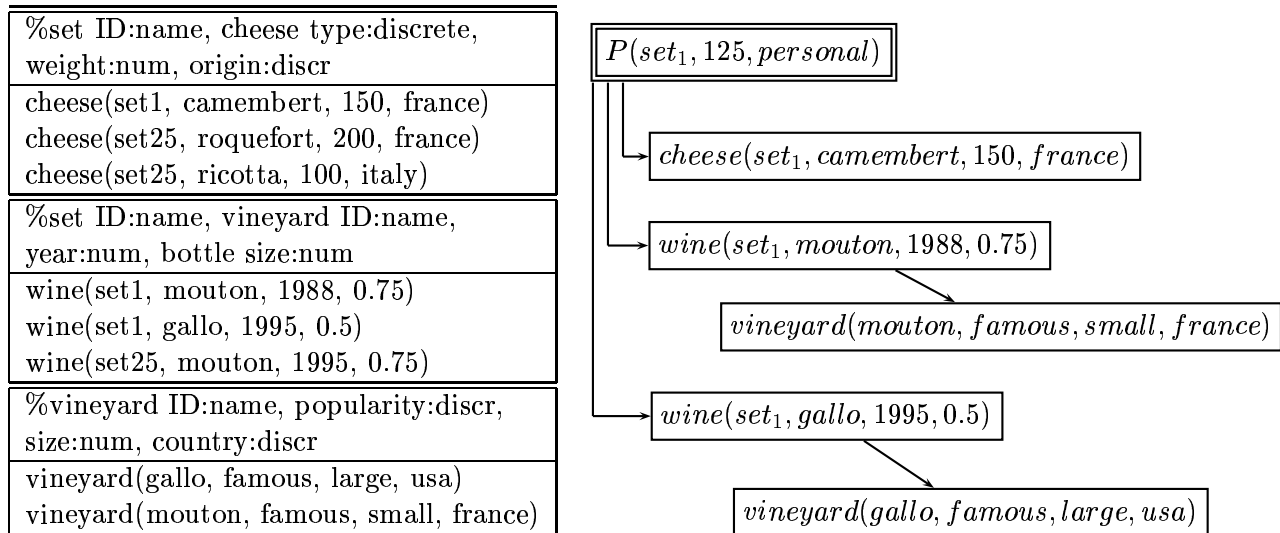


Figure 3.3: On the right, the case defined by I with the respect of the background knowledge \mathcal{B} given on the left. The depth is $d = 2$.

in the next paragraph.

3.5.1 First Order Distance Measure

One of main goal of first order relation mining is to provide suitable algorithms to calculate distance between two first order instances $x, y \in X$. The key idea is to compute the distance by recursively comparing the components of first-order instances until one either falls back to the prepositional comparison on elementary features or a maximum depth bound is reached. Recalling again the example provided in [2], let's suppose we have two instances $I_1 = P(set1, 125, personal)$ and $I_2 = P(set25, 195, mail)$ with the background knowledge of fig. 3.3. We want to define a distance between I_1 and I_2 .

It is intuitive to fix a depth bound d and to compute the distance as the normalized sum of the distances between the instances' corresponding arguments. As a consequence, $dist$ is a function $dist : \mathcal{X} \times \mathcal{X} \times \mathcal{B} \times INT \rightarrow [d_{min}, d_{max}]$ computed recursively with the respect to the cases derived by the given background knowledge \mathcal{B} . More precisely, we have (fig.3.4)

$$dist(I_1, I_2, \mathcal{B}, d) = \frac{1}{3}(dist(set_1, set_2) + dist(125, 95) + dist(personal, mail))$$

With the respect with non-relational case, the most important modification introduced is the introduction of a methods to compute the distance $dist(set_1, set_2)$ between the object identifiers set_1 and set_{25} . We do this, taking in account the case of set_1 and set_{25} , and partitioning them by the predicate symbols. We obtain the sets $L_{set_1, wine}$, $L_{set_1, cheese}$, $L_{set_{25}, wine}$,

and $L_{set_{25},cheese}$. To compute the distance D_{cheese} we note that the cardinality of $L_{set_1,cheese}$ is one and the cardinality $L_{set_{25},cheese}$ is two. Therefore, we can compute the distance between the two sets as the minimum among all the elements in them, normalized with the respect of the larger set $L_{set_{25},cheese}$. A similar computation is done for D_{wine} between $L_{set_{25},wine}$ and $L_{set_1,wine}$. Here we need to compute also the distance $dist(mouton, gallo)$, since they are not ground atoms. Having D_{wine} and D_{cheese} it is possible to calculate $dist(set_1, set_{25})$ and to obtain $dist(I_1, I_2, \mathcal{B}, d)$. Formally, we have the following definition:

Definition 3.1 *A **first order relational distance** between two instances $I_1, I_2 \in \mathcal{X}$ is computed having a vocabulary associated with an argument type and mode declaration for each of its predicate symbols, having a background knowledge \mathcal{B} consisting of ground atoms (they contain no variables) and having a depth bound $d \geq 0$ which controls the recursion. First at all, the case of I_1 and I_2 , with the respect to \mathcal{B} and d , is computed. Then, to compare I_1 and I_2 a recursive visit is performed, according to the computed case. There are different strategies according to the type of the arguments:*

1. **number and discrete:** a corresponding elementary distance metric is called
2. **list and term:** the metric is based on the edit distance, which we will define in section 3.6.1.
3. **name:** This kind of type exploits recursion up to the depth bound d . If d is reached the arguments are considered as discrete and the metric of (1) is used. Otherwise, the cases of the respective named objects are collected. We obtain two sets that are partitioned by predicate symbols. Every predicate symbol is compared with the corresponding subsets from the two sets: for each fact from the set with smallest cardinality we compute recursively its distance from the larger set, then sum up these distances and normalize them with respect of the larger set.

$$\begin{aligned}
I_1 &= P(\text{set1}, 125, \text{personal}) && \text{(first instance)} \\
I_2 &= P(\text{set25}, 195, \text{mail}) && \text{(second instance)} \\
\text{dist}(I_1, I_2, \mathcal{B}, d) &= \frac{1}{3}(\text{dist}(\text{set}_1, \text{set}_2) + \text{dist}(125, 95) + \text{dist}(\text{personal}, \text{mail})) && \text{(relational distance)} \\
\text{dist}(125, 195) &= |195 - 125| / 500 = 0.14 && \text{(500 is the highest price)} \\
\text{dist}(\text{personal}, \text{mail}) &= 1 && \text{(since they are different)} \\
\text{dist}(\text{set}_1, \text{set}_{25}) &= \dots && \text{(need to recurse up to } d) \\
L_{\text{set}_1, \text{wine}} &= \{ \text{wine}(\text{set}_1, \text{mouton}, 1988, 0.75), \text{wine}(\text{set}_1, \text{gallo}, 1995, 0.5) \} && \text{collect sets of literal} \\
L_{\text{set}_1, \text{cheese}} &= \{ \text{cheese}(\text{set}_1, \text{camembert}, 150, \text{france}) \} && \text{and partition by the} \\
L_{\text{set}_{25}, \text{wine}} &= \{ \text{wine}(\text{set}_{25}, \text{mouton}, 1995, 0.75) \} && \text{predicate symbol} \\
L_{\text{set}_{25}, \text{cheese}} &= \{ \text{cheese}(\text{set}_{25}, \text{roquefort}, 200, \text{france}), \text{cheese}(\text{set}_{25}, \text{ricotta}, 100, \text{italy}) \} \\
D_{\text{cheese}} &= \frac{\min_{l \in L_{\text{set}_{25}, \text{cheese}}} \text{dist}(\text{cheese}(\text{set}_1, \text{camembert}, 150, \text{france}), l)}{|L_{\text{set}_{25}, \text{cheese}}|} && \text{(minimum distance)} \\
&= \frac{\min\{(1 + \frac{|150-200|}{300} + 0)/3, (1 + \frac{|150-100|}{300} + 1)/3\}}{2} \\
&= 0.1944 \\
\text{dist}(\text{mouton}, \text{gallo}) &= \frac{\min(\text{dist}(\text{famous}, \text{famous}) + \text{dist}(\text{small}, \text{large}) + \text{dist}(\text{france}, \text{usa}))/3}{1} \\
&= 0.6666 \\
D_{\text{wine}} &= \frac{\min_{l \in L_{\text{set}_1, \text{wine}}} \text{dist}(\text{wine}(\text{set}_{25}, \text{mouton}, 1995, 0.75), l)}{|L_{\text{set}_1, \text{wine}}|} && \text{(minimum distance)} \\
&= \frac{\min\{(0 + \frac{|1995-1988|}{100} + 0)/3, (0.6666 + 0 + \frac{|0.75-0.5|}{1})/3\}}{2} \\
&= 0.0117 \\
\text{dist}(\text{set}_1, \text{set}_{25}) &= \frac{\min(D_{\text{cheese}} + D_{\text{wine}})}{2} = 0.1031 \\
\text{dist}(I_1, I_2, \mathcal{B}, d) &= \frac{0.1031 + 0.14 + 0}{3} = 0.081
\end{aligned}$$

Figure 3.4: An example of first order relational distance between two instances I_1 and I_2 . The background knowledge \mathcal{B} is the same of the previous example and $d = 2$.

3.6 Word based Relational Model

In this simplified relational model objects are simply a set of words $\{w_1, \dots, w_m\} \subseteq \mathcal{O}$, instead of a certain number of first order predicates. It is still possible to define relations among objects and to introduce the notion of distance, as we will do in the next section.

3.6.1 Edit Distance and Relational Mining

The edit - or Levenshtein distance - is another kind of relational distance. The distance determines the number of edit operations (delete, insert, change) needed to convert one word into an other. It is defined as it follows:

$$L((x_1, \dots, x_p), (y_1, \dots, y_q)) = \begin{cases} q & p = 0 \\ p & q = 0 \\ \min(L((x_1, \dots, x_{p-1}), (y_1, \dots, y_q)) + 1, \\ L((x_1, \dots, x_p), (y_1, \dots, y_{q-1})) + 1, \\ L((x_1, \dots, x_{p-1}), (y_1, \dots, y_{q-1})) + z(x_p, y_q) & o.w. \end{cases}$$

where

$$z(x, y) = \begin{cases} 0 & x = y \\ 1 & o.w. \end{cases}$$

It is possible to compute L recursively or iteratively, using dynamic programming. Moreover, one can assign different weights to delete/insert/change operations. Edit distance is largely used in Web Relational mining as we will see in section (3.6). An example of iterative computation is given in fig. 3.5.

| L_{ij} | | c | l | u | s | t | e | r | |
|----------|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| c | 1 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| l | 2 | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 |
| a | 3 | 3 | 2 | 1 | 1 | 2 | 3 | 4 | 5 |
| s | 4 | 4 | 3 | 2 | 2 | 1 | 2 | 3 | 4 |
| s | 5 | 5 | 4 | 3 | 3 | 2 | 2 | 3 | 4 |

Figure 3.5: Example of dynamic programming: the edit distance between `cluster` and `class` is 4

3.7 Relational Mining Applications

In the next sections, we present some examples of applications developed using the notion of first order distance or the notion of edit distance both on different kind of relational data sets.

3.7.1 First Order Mining Applications

With the notion of relational distance embedded in an instance space \mathcal{X} , it is possible to extend to the first order several mining methodologies. In the next sections we report the description of RIBL2 a predictive instance learner, RDBC a hierarchical agglomerative clustering, and FORC a relational k -means like clustering algorithm. All these are presented in [2].

3.7.1.1 Relational Instance Based Learning

Given a repository of examples \mathcal{E} and a new relational instance $q \in \mathcal{X}$, we identify the set of $N \subseteq \mathcal{E}$ which are the k nearest neighbors of q . We have two different cases:

1. If we are predicting a discrete variable $h(x_{new})$ then we use the function

$$h(x_{new}) = \underset{y_j}{\operatorname{argmax}} \sum_{e=(x,y_j) \in N_{y_j}} \operatorname{sim}(x, x_{new})^\alpha$$

where $N_{y_j} = \{e = (x, y_j) \in N\}$ is the set of those neighbors that are voting for value y_j , $\operatorname{sim}(x, y) = 1 - \operatorname{dist}(x, y)$, and α is a parameter which determines how much influence has the farther-away neighbors.

2. if we are predicting a continuous variable, then we use a weighted average function defined as it follows:

$$h(x_{new}) = \frac{\sum_{e=(x,y_j) \in N_{y_j}} \operatorname{sim}(x, x_{new})^\alpha \cdot y_j}{\sum_{e=(x,y_j) \in N_{y_j}} \operatorname{sim}(x, x_{new})^\alpha}$$

In [2] Kirsten et al. presents RIBL2 a predictive learner which classifies relational instances in a lazy way, according to the above formulation. It is used with success in the genomic domain analysis.

3.7.1.2 Relational Hierarchical Agglomerative Clustering

In [2] Kirsten exploits the relational distance for performing Hierarchical clustering. This approach is used in RDBC software. It starts with the list of cases as one-element clusters, and iteratively groups the two most similar clusters in a new cluster. The just merged clusters are removed and the union is inserted in the set of “alive clusters”. Joining two nearest clusters, the distance between the new created cluster and all the remaining can be computed from the already known distances of current clusters. The general rule is the following:

$$\text{dist}(C_1 \cup C_2, C_3) = f(\text{dist}(C_1, C_3), \text{dist}(C_2, C_3))$$

if $f = \min$ we have the class of algorithms known as *single link clustering*, if $f = \max$ we have the *complete-link clustering*, and if $f = \text{average}$ the *average-link clustering*. RDBC can implement all three variants.

Hierarchical Clustering produce a dendrogram, RDBC automatically “cuts” it at a certain level of depth such that the quality of the cluster is maximized. This is obtained minimizing the intra-cluster distance. It is defined a measure of clusters’ compactness $\text{qual}(C) = \frac{\sum_{c \in C} \text{sim}_{\text{intra}}(C)}{|C|}$, where $\text{sim}_{\text{intra}}(C)$ is the intra-cluster similarity of a cluster C .

Note that this measure is minimized having clusters composed of just an element. To avoid this pathological case, single element clusters are moved to a special *outlier cluster*, and the optimization of $\text{qual}(C)$ is performed on the resulting clusters’ assignment as it follows. Let $SIM = \{\text{sim}_{\text{intra}}(C) | C \in \mathcal{C}\}$ and for any element $s \in SIM$, let

$$C(s) = \{C \in \mathcal{C} | \text{sim}_{\text{intra}}(C) \geq s \text{ and } \text{sim}_{\text{intra}}(\text{parent}(C)) < s\}$$

The chosen clustering is $C_{\text{chosen}} = \text{argmax}_{s \in SIM} \text{qual}(C(s))$.

RDBC was successfully [2] used for performing relational clustering in the mutagenesis genetic domain, achieving better accuracies than the previous non-relational approaches.

3.7.1.3 Relational K-means

A relational distance can also be exploited to extend the k -means clustering algorithm to the first order computation. Particular attention should be paid to generation of the center of a set of case in \mathcal{X} , since in this case the notion of “center” has no intuitive geometric interpretation as in the non-relational k -means. Remember that, in non relational context, the centroid is simply $\text{center}(C) = \frac{1}{|C|} \sum_{c \in C} c$.

A first solution proposed is to take, as approximation of the center, an already existing predicate in the instance space \mathcal{I} . This approach is used in [2] and in [5] and it is named, in non-relational literature, as *k-medoids*. In [2] Kirsten present FORC, a software which

implements relational clustering using a first order K -means algorithm and k -medioids. From a computational point of view, the work done is proportional to the squared number of instances in a given clusters.

A different approach - named *k-prototypes* - was proposed in [5] by Kirsten and Wrobel. They suggest using a heuristic to create a new “center” instance prototype, given a set of relational instances. The goal is create a new instance N which minimises the squared distance between N and all the already existent instances $\mathcal{I} = \{I_1, \dots, I_n\}$ in the specific cluster. The creation is based on a recursive computation which takes in account the given instances and the background knowledge \mathcal{B} . Each instance I_j consists of a target fact F_j and the set of related facts in \mathcal{B} . The computation works as it follows. Starting with $F = \{F_1, \dots, F_n\}$ with a given predicate symbol and arity q , the prototype is generated with a new q -ary atom and iteratively calculating the values of its arguments:

- If an argument has an atomic type of number or constant, the computation is based on propositional representation. In this case the distance of two symbolic arguments

$$a_1 \text{ and } a_2 \text{ is } d(a_1, b_1) = \begin{cases} 0 & a_1 = a_2 \\ 1 & o.w. \end{cases} .$$

- If the argument is neither a number nor a constant, then for all objects referred by the argument a new sub-prototype is build recursively. The recursion depth is bounded by a given depth bound d . In [5] this approach is extended to manage lists. Two different solutions are evaluated to build an “*average list*”. The first one, more expensive, exploits the edit distance. The second one is based on a majority vote mechanism. A detailed example of prototype generation is provided in Appendix A of [5].

The experimental evaluations, reported in [5] on the mutagenecy genetic domain, suggest that k -medioids outperforms the quality results given by the k -prototypes, especially for medium or large values of k . The authors argue that this bad performance is a consequence of the “loss” of clusters. At the iteration $i + 1$, the prototypes $\{N_1, \dots, N_k\}$, generated at iterations i , have very often assigned no instances to their clusters. So, from time to time, the number of clusters with no population will increase. Another inefficiency showed by the k -prototype approach is that it tends to go into cyclic states, instead of converging in few iterations to a (local) optimum. In conclusion, Kirsten suggest to adopt a k -medoid approach instead of a k -prototype one for the mutagenicity domain.

3.7.2 Word based Relational Applications

In this section, we focus our attention to the applications presented in literature that exploit a relational model based on words as ground object. It is successfully used in Web mining domain. For instance, in [6] Runkler and Bezdek suggest to use relational mining for:

- Clustering web access logs files that contain access sequences of web pages visited by users.
- Clustering web pages themselves, exploiting their textual content and the HTML structure used for composing them.

We will describe in detail such applications in the next sections.

3.7.2.1 Relation Web Mining Applications

The access paths to a Web server can be described using a graph $G = (N, E)$ where N are the set of pages, and there exist an edge $(i, j) \in E$ if an users coming from page i visit a page j . A first kind of distance defined in [6] is the *graph distance*:

$$D(G_1, G_2) = 1 - \frac{2||E(G_1) \cap E(G_2)||}{||E(G_1)|| + ||E(G_2)||}$$

where $D(G_1, G_1) = 0$ and $D(G_1, G_2) = 1$ if $G_1 \cap G_2 = 0$. Note that this distance does not consider the order of nodes visited. A different solution to take in account the *sequence* of nodes visited is to mape to a symbol each node in the sequence. Therefore, different paths are mapped to strings and their distance can be measured using the edit distance. Another solution, which mixes the above two models, is to use the edit distance and replace, in its definition, the $z(x, y)$ with the distance between nodes x and y in the graph (i.e. the number of edges in the shortest path between x and y).

In [6] the clustering problem is seen as the optimization problem of the objective function:

$$J(U, V; X) = \sum_{i=1}^c \sum_{k=1}^n u_{ik} ||x_k - v_i||^2$$

where $X = x_1, \dots, x_n$ is the data set, which is partitioned in $c \in 2, \dots, n - 1$ clusters, U is a partition matrix $c \times n$ and V contains c clusters prototypes. It holds that $\sum_{i=1}^c u_{ik} = 1, \forall k = 1, \dots, n$ and $\sum_{i=1}^c u_{ik} > 0 \forall k = 1, \dots, c$. It can be shown that the above equation can be minimized alternatively computing U and V as it follows (until a stopping criterion is reached):

$$u_{ik} = \begin{cases} 1 & ||x_k - v_i|| = \min_{j=1, \dots, c} \{ ||x_k - v_j|| \} \forall i, k \\ 0 & o.w. \end{cases}$$

where

$$v_i = \frac{\sum_{k=1}^n u_{ik} x_k}{\sum_{k=1}^n u_{ik}} \quad \forall i$$

In [6] were reported fuzzy and possibilistic extension to the above method. All of them can produce clusters alternating the computation of $U(V, X)$ and $V(U, X)$ using the equations

derived from the necessary conditions for optima in $J(U, V; X)$. This kind of algorithms are called *alternating optimization* (**AO**).

A slight different approach, named *alternating cluster estimation* (**ACE**), requires that the user provides directly the desired membership function $U(V, X)$ and/or the desired prototype function $V(U, X)$. So, the difference is that these two functions are user-provided instead of analytically derived by the necessary optima conditions on $J(U, V; X)$. For instance, the authors suggest using:

$$v_i = \frac{\sum_{k=1}^n u_{ik}^\lambda x_k}{\sum_{k=1}^n u_{ik}^\lambda} \quad u_{ik} = \begin{cases} 1 - \frac{\|x_k - v_i\|^\alpha}{r_i} & \|x_k - v_i\| \leq r_i \\ 0 & o.w. \end{cases} \quad (\alpha, \lambda, r_1, \dots, r_c) \in \mathcal{R}^+$$

When ACE is extended to a relational context, it is named *relational alternating cluster estimation* (**RACE**). RACE randomly takes c initial cluster centers V from the instance space \mathcal{X} , and then alternatingly computes the membership matrix U and the cluster centers V . Runkler and Bezdek suggest to use the edit distance and the following values for $U(X, X)$.

$$u_{ik} = \frac{1}{1 + d_{ik}} \quad i = \{1, \dots, c\} \quad k = \{1, \dots, n\}$$

and to adopt a maximum operator to update $U(V, D)$. This operator randomly takes the i -th cluster at step j and find the object o_k in \mathcal{X} that has the maximum membership in cluster i . Then it sets $v_i = k$.

3.7.2.2 Clustering and Keyword Extraction on Usenet News

RACE was used to cluster articles belonging to 20 newsgroups. These discussion groups were taken from Usenet and stored at Carnegie Mellon University. 20000 articles composed the dataset, 1000 for each of the 20 groups. A clustering computation, based on the edit distance, was done for each discussion group. The center of each cluster is represented by a label composed by a single word. The authors did not report any standard measure [1, 4] to identify the quality of results (such as precision and recall, entropy, intra-cluster and inter-cluster distance). Nevertheless, they notice that the labels generated are meaningful. This is a very good result obtained just using edit distance, with no semantic knowledge.

3.7.2.3 Classification of Articles

The above keyword extraction method can be used as a feature selection tool. Then, a classification process can be realized using the edit distance. More precisely, each article is represented by a sequence of words (w_1, \dots, w_m) and each word w_j has a minimum distance to one of the cluster centers $v_i^{(l)}$ where $i = 1, \dots, 20$ on each newsgroup $l = 1, \dots, 20$. Using

$$L'_i = \sum_{j=1}^m \min_{\{i=1, \dots, c\}} \{L(w_j, v_i^{(l)})\}$$

where L is the edit distance, the classification assigns articles to the group with minimize the distance L'_i . The authors report a classification precision ranging from 60% to 81% according to the selected discussion group.

3.7.3 Web log mining

The edit distance or the graph distance defined in the above paragraphs can be used for mining web logs. The authors did an experiment on 34299 accessed pages taken from Rensselaer Polytechnic institute. Again, the edit distance was used to select the cluster center and to measure distance. The number of clusters was fixed to $c = 5$ and they adopted the following fuzzy membership function of x in the i_{th} cluster:

$$u_i(x) = \begin{cases} 1 / \sum_{j=1}^5 \frac{L(x, v_i)}{L(x, v_j)} & i = 1, \dots, 5 \\ 1 & L(x, v_j) \neq 0 \quad \forall j = 1, \dots, 5 \\ & L(x, v_i) = 0 \rightarrow \text{assign } u_j(x) = 0 \quad \forall j \neq i \end{cases}$$

where $u_i(x) \in [0, 1]$ and $\sum_{i=1}^5 u_i(x) = 1$.

The authors report clustering results which seems intuitive but do not report any kind of measure of quality.

3.8 Conclusion

In this lecture, we have presented the state-of-art applications of Relational Mining. Relational Mining is a new and valuable mining paradigm where objects are described in such a way that is more inherent to their own real nature. Denoting objects by means of words or first order predicates and giving relations among them, can produce a more accurate description of the real world phenomena. Therefore, all mining techniques (such as clustering, classification, predictive learning and so on) can potentially give results that are more precise. At first view, it could be somewhat surprising that this paradigm-shift was reached just defining a relational distance. But, several mining techniques can be easily described using a distance (Euclidean or not) on some m -dimensional space. Therefore, defining a relational distance allows us to easily extend these techniques on a non-numeric space.

Drawbacks Nevertheless, there are some drawbacks. First at all, relational distance computation can be very expensive. For instance, the cost of computation of the edit distance $L(s_1, s_2)$ is $O(|s_1| * |s_2|)$, while the cost of computation of a first order distance depends on

the chosen depth bound d which controls recursion and on the variety of predicates. This poses serious scalability problems. In particular, for Web mining applications where one can assume an high-order dimension derived by the huge number of features existing in the data sets to mine.

Besides, relational space could be a non-continuous space. This has some implications on those methods that needs to compute a “center” that represents a cluster. Solutions like k -medoids seems viable, but solutions like k -prototype produced (very) bad results. Moreover, the lack of continuity can have some impact to convergence. Methods such as K -means could not find a (local) optimum and continuing to oscillate in cyclic states.

Another aspect to take in account is that the relational representation is more space-greedy. As an example, consider that the instance-based learning methods must store all instances until prediction/classification time. This is more space consuming than storing a bitmap or a vector.

Last but not least, first order measures given in literature can compute distances only when the background knowledge \mathcal{B} is just composed by ground atoms, i.e. contains no variables.

However, first order relational mining was successfully used in genetic domain and in several other contexts (such as pollution analysis, bio-chemistry, natural language processing, traffic control and others). Instead, in Web Mining context the cheaper word-based relational model is a good compromise between precision, derived by the use of non-numeric objects, and performance. We have reviewed some examples of clustering, distance based classification and web logs mining.

Future Researches There is still room for further researches. The first obvious direction is to find methods that alleviate the cost of distance computation.

Another line is to provide extensive testing about the quality of the obtained results. This should be done measuring precision, recall, entropy, inter and intra cluster distance on standard corpus and on synthetic datasets.

Other lines of research consist in identifying different applications for the relational methodologies. A modest proposal is to investigate the use of relational mining techniques in the information retrieval context. In the following list, we identify some examples of web relational application for search engines:

- **Query Spelling.** In this problem, objects are *misspelled words* and the task is to suggest the correct spelling. The dataset to mine is the lexicon built by the search engine at the indexing time. It is natural to model it considering as ground objects the words themselves and to describe the relations between words in terms of a relational distance. For instance, Google provides this kind of service.
- **Query Refinement.** Another context where one can use a relational model, is when

one want to suggest to the users possible refinements for the queries they have submitted.

- **Automatic Query Generation.** This is very recent line of research in Information Retrieval. The goal is to automatically extract a query Q from a given (set of) document(s) and to submit automatically Q to a News/Web search engine to retrieve the most recent information about the topics described in the document(s). This is somewhat related to the automatic keyword extraction suggest in the above paragraphs.
- **News/Web Blogs Clustering.** Another very interesting direction of research could be the use of relational models for automatic clustering of News or Web Blogs. This in order to provide to the users the freshest news about a set of topics she has selected.

References

- [1] Soumen Chakrabarti. *Mining the Web, Discovery knowledge from hypertext data*. Morgan Kaufmann, 2003.
- [2] M Kirsten and S Wrobel. Relational distance-based clustering. In Springer-Verlag, editor, *Proceedings of the 8th International Conference on Inductive Logic Programming, volume 1446 of Lecture Notes in Artificial Intelligence*, pages 261–270, 1998.
- [3] Mathias Kirsten, Stefan Wrobel, and Tamas Horvath. *Distance Based Approaches to Relational Learning and Clustering*. SpringerVerlag, 2001.
- [4] C.D. Manning and H Schutze. *Foundations of Statistical Natural Language Processing*. MIT Press, 2002.
- [5] Stefan Wrobel Mathias Kirsten. Extending k-means clustering to first-order representations. *ILP*, pages 112–129, 2000.
- [6] J.C. Bezdek T.A. Runkler. Web mining with relational clustering. *Elsevier*, pages –, 2001.